(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 387 226 A1**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 90850095.2

(51) Int. Cl.5: G06F 15/38, G06F 15/40

(22) Date of filing: 05.03.90

(30) Priority: 06.03.89 SE 8900774

(43) Date of publication of application:
12.09.90 Bulletin 90/37

(84) Designated Contracting States:
AT BE CH DE DK ES FR GB GR IT LI LU NL SE

(71) Applicant: International Business Machines Corporation
Old Orchard Road
Armonk, N.Y. 10504(US)

(72) Inventor: Hedin, Erik Bertil
Kostervägen 10 n.b.
S-181 35 Lidingö(SE)
Inventor: Jonsson, Gregor I.
Tallstigen 3
S-818 62 Lidingö(SE)
Inventor: Olsson, Lars Erik
Sorögatan 19
S-164 41 Kista(SE)
Inventor: Sanamrad, Mohammad A.
Lojovägen 52
S-181 47 Lidingö(SE)
Inventor: Westling, Sven Olof Gunnar
Falkstigen 79
S-182 75 Stocksund(SE)

(74) Representative: Johansson, Lars E. et al
IBM Svenska AB Intellectual Property Department 4-01
S-163 92 Stockholm(SE)

(54) Natural language analyzing apparatus and method.

(57) A system for natural language (NL) analyzis is provided which comprises apparatus and method. It has the capability to analyze NL expressions and to resolve ambiguities and present them to the user for verification of correct interpretation.

The invention is based on a language independent model of the contents of a data base. The model contains records of information defining entity types and relations between such entity types. The model is created (customizing the system) by the user, and is stored as a conceptual schema.

The entities of the schema are linked or connected to natural language terms in a vocabulary.

The entities represent objects in the data base, and there is a link or connection between the entities and those objects of the data base.

The actual analysis of NL expressions is performed by a Natural Language Engine (NLE) in cooperation with an Analysis Grammar and the schema. The analysis results in an intermediate, language independent logic form representation of the input, which is paraphrased back to NL for verification, and if the input is a query, there is a translation into a query language such as SQL.

EP 0 387 226 A1

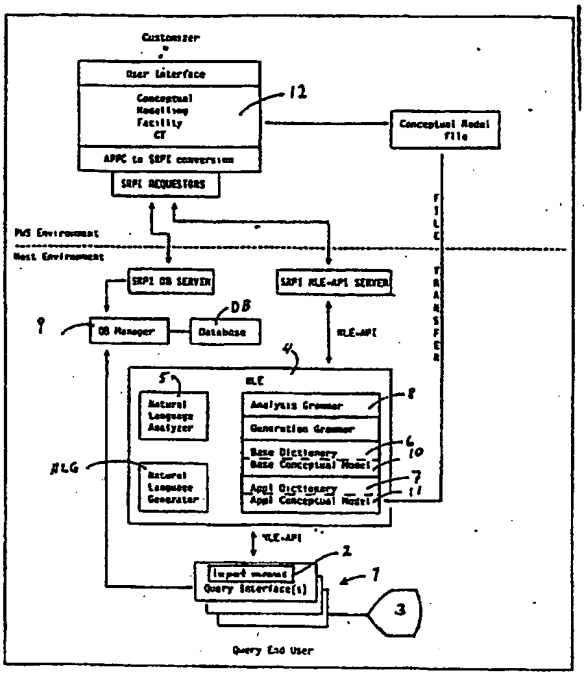
Customizer
User Interface
Conceptual Modelling Facility CF
APPC to SRPI conversion
SRPI REQUESTORS
12
Conceptual Model File
FILE TRANSFER
PWS Environment
Host Environment
SRPI DB SERVER
SRPI NLE-API SERVER
DB Manager
Database
DB
NLE-API
NLE
Natural Language Analyzer
Analysis Grammar
Generation Grammar
Base Dictionary
Base Conceptual Model
Appl Dictionary
Appl Conceptual Model
NLG
Natural Language Generator
Input manager
Query Interface(s)
Query End User

Fig. 1

NATURAL LANGUAGE ANALYZING APPARATUS AND METHOD

The present invention relates in general to the use of natural language for communication with computers, and in particular to querying data bases, e.g. relational data bases, or to translation between two natural languages of application specific texts.

There is a widely recognized demand in the computer world for user friendly interfaces for computers. Numerous attempts have been made in order to achieve this with various results.

The simplest way of creating programs that are possible to use without having particular skills is to design menu based systems where the user selects functions from a panel with several options.

Another way is to make use of screens with symbols ("icons") and letting the user select from the screen by pointing at the selected symbol with a light-pen, or by moving a cursor by means of a so called "mouse", pointing at the desired symbol, and then pressing a button for activating the function.

These methods have severe limitations in many applications where great flexibility in selection is desired, since such systems must be predefined, and unexpected or new desires requires programming of the system again.

Especially for data retrieval from data bases the need for flexibility is evident. In order to make searches in data bases, often complex query languages must be used, requiring high skill, and if reports are to be created from the retrieved data, further processing must be carried out. In addition several succesive queries may have to be entered before the end result is arrived at.

An example of a query language is SQL (Structured Query Language; IBM program no 5748-XXJ). This is widely used but due to its complexity it is not possible for the average user to learn it satisfactorily, instead there are specialists available for creating SQL-query strings that can be implemented as commands for searches of a routine nature. The specialist must be consulted every time a new kind of query is to be made.

There have been numerous attempts to remedy such deficiencies by trying to create interfaces to data bases which can interpret a query formulated in natural language. However, practically every such attempt has been based on key word identification in the input query strings. This inevitably leads to ambiguities in the interpretation in many cases.

Rather recently the research in the artificial intelligence area has led to systems where lexical, syntactical, and semantic analysis has been performed on input strings, utilizing grammars and dictionaries, mainly for pure translation purposes. It seems as if these systems are succesful only to a certain extent, in that there is a relatively high rate of misinterpretations, resulting in incorrect translations. This frequently leads to the requirement of editing of the result.

GB-2 096 374-A (Marconi Company) discloses a translating device for the automatic translation of one language into another. It comprises word and syntax analysis means, and the translation is performed in two steps by first translating the input sentence into an intermediate language, preferably artificial, and then translating the intermediate language into the target language.

EP-0 168 814-A3 (NEC Corporation) discloses a language processing dictionary for bidirectionally retrieving morphemic and semantic expressions. It comprises a retrieving arrangement which is operable like a digital computer, and the dictionary itself is comprised of elementary dictionaries, namely a morphemic, a semantic and a conceptual dictionary. Each morphemic and conceptual item in the corresponding dictionaries are associated with pointers to a set of syntactical dictionary items. The syntactical items are associated with two pointers to a set of morphemic and a set of conceptual items.

US-4 688 195 (Thompson et al, assignee Texas Instruments) discloses a natural language interface generating system. It generates a natural language menu interface which provides a menu selection technique particularly suitable for the unskilled user.

However, none of the above listed patents fully adresses the problem solved by the present invention, although they do present alternative technical solutions to certain features.

The object of the present invention is to provide a device and a method by means of which a user can formulate input expressions in a selected natural language in reasonably random fashion, which expressions are interpreted lexically, syntactically, and semantically by means of dictionaries and analysis grammars, ambiguities of said expressions are resolved, and the expressions are transformed into an intermediate representation form.

The intermediate representation form can then be used for creating queries in a specific query languge (such as SQL for a relational data base).

The present invention as defined in the appended claim 1, will achieve the above mentioned object.

The systems interpretation is paraphrased and a "play-back" of the input is presented to the user for

verification of the correctness of the interpretation of the input expression or query. For this purpose there may be provided a natural language generator including a generation grammar.

If the generation grammar is for another natural language than that of the analysis grammar, the latter function can also be used for pure translation into another language.

The invention is based on a language independent model of the contents of a data base, in the form of records of information defining entity types and relations between such entity types. The entities denote the concepts behind the data in the data base. Such a collection of records is in the art of conceptual modelling referred to as a "conceptual schema".

The entities in such a collection of records or "conceptual schema" are connected to natural language terms in a vocabulary. The schema itself is completely language independent, and contains only the entities (concepts) and relations between entities.

By using such a schema, which is a model of a relevant so called 'Universe of Discourse' (or object system, which is a collection of abstract and/or concrete things and information about these things, to which the natural language expression to be analyzed, is relevant), it is possible to obtain complete resolution of ambiguities, as long as the input expression is in reasonable agreement with the Universe of Discourse. This has not been possible previously.

Since the schema is language independent there is a great advantage in that it is very easy to change analysis grammar and vocabulary, and thus to switch between different natural languages. In fact grammars and vocabularies can be supplied as 'plug-in' modules.

In a preferred embodiment of the invention the schema is also connected to the contents of a relational data base. That is, each concept of the schema may or may not have a unique connection to a table containing objects (data) relating to that concept.

Thus, the schema constitutes a link between natural language and the data base. If thus the input expression is a query to the data base, the analysis will produce an interpretation of the query which then is translated into the query language for that data base (e.g. SQL).

In another embodiment, queries are paraphrased, i.e. if a query is ambiguous, two or more paraphrases are presented to the user, for him to select the correct one. Thereby one achieves that a correct query is made to the data base.

In a further embodiment the paraphrasing function is used for pure translation. Thereby a generation grammar and a vocabulary for a second language is used when paraphrasing the input expression. In this case there is no use of a data base in the sense of the previously mentioned embodiment.

In the following, preferred embodiments of the invention are disclosed in the detailed description of the invention given below, with reference to the drawings, in which

fig. 1 is an overview of a system comprising the natural language analyzing device according to the present invention, as implemented for querying a relational data base,

fig. 2A is a schematic illustration of a simple example of a conceptual schema, modelling the data base contents, and which can be used with the invention,

fig. 2B is a simplified illustration of how parts of the schema of fig. 2A are linked to tables in a data base and to natural language terms in a vocabulary,

fig. 3A is an example of a parse tree (or syntax tree) created during parsing,

fig. 3B is a graphic illustration of a semantic tree built by the parser, and

fig. 4 is an illustration of the screen of the graphic interface of the Customizing Tool.

With reference now to fig. 1, the general lay-out and design of a system for querying a data base comprising the invention will be given.

A data base query system incorporating the invention thus has a Query Interface 1 comprising Input Means 2 that can have any suitable form for transforming character strings into digital signals, e.g. a keyboard of standard type. It is also conceivable that the input query is made by speech, in which case the input means would comprise a microphone and sound analyzing means.

There may also be present a Display Means 3 for presenting results of queries, results of parsing (paraphrased queries; to be described later), and also for displaying e.g. help panels.

The core of the system is the Natural language Engine NLE 4. It comprises a Natural Language Analyzer 5 which includes a Parser and which is used for the actual syntax analysis. The Analyzer makes use of dictionaries 6,7 (Base Dictionary and Appl Dictionary) and an Analysis Grammar 8 to perform the actual parsing of the input (to be described in more detail later).

The system further comprises a Data Base (DB) manager 9. It will not be described in detail since the man skilled in the art readily recognizes the necessary design of such a device.

An essential feature of the invention is a model of the data base in the form of a conceptual schema (Base Conceptual Model 10 and Appl Conceptual Model 11), which may be created by the user.

The concept of a conceptual schema is described in the litterature in the field of artificial intelligence, (see e.g. "Konceptuell Modellering" by J. Bubenko et al).

Briefly, a conceptual model consists of

1) 'Entities', which are any concrete or abstract thing/things of interest;
2) 'Relationships' which are associations between entities;
3) 'Terms' which are natural language expressions that refers to entities;
4) 'Database Representations' which are mappings of entities into the database; and
5) 'Database Information' comprising 'Referential Integrity' and 'Key'

As many entities as the user finds necessary may be defined, and the system will automatically suggest that every table in the data base is associated with an entity.

Entities of the model may be connected or linked to each other by one or several relationships. In general relationships fall into the following categories:

'is an instance of'
'identifies'
'is named'
'is a subtype of'
'is counted by'
'is measured by'
'possesses'
'subject'
'direct object'
'dative object'
'preposition'
'adverbial of place'
'adverbial of time'
.
.
.
(etc)

The 'subtype' relationship is a hierarchical relationship and is treated separately from the other non-hierarchical relationships. Most of the above relationships are self-evident as to their meaning, but for clarity a few examples will be given (see Fig 2A and 2B):

CNTRY(entity;e2) 'possesses'(relationship)
CPTL(entity;e1)
PRDCR(e3) 'is subtype of' CNTRY(e2)
EXPORT(e6) 'has object' PRDCT(e8) /in this example the entity EXPORT has no link to a table in the data base/

Entities of the model are connected to natural language terms by the user, apart from a base collection of terms common to all applications (e.g. list, show, who, what which, is, more etc.). Such terms are members of a base dictionary which is part of the system initially. It should be noted that an entity may be associated with zero, one or more natural language terms of the same category. The same term can also be associated with more than one entity.

The actual building of the model, comprising connecting it to the natural language terms and to tables of the data base, is performed with a Customization Tool (CT) 12 (described later). The "SRPI"-boxes denote what one might call a communication protocol, necessary for communication with the host, for accessing the data base during customization (SRPI = Server Requester Programming Interface).

The way in which the conceptual model has been used to form a natural language interface to a data base or for translation purposes by connecting it to natural language terms is not previously disclosed.

With reference now to fig. 2A and 2B an example of how the conceptual model is implemented within the scope of the invention will be given. In the example a relational data base with tables containing information about a number of countries, is assumed as the information containing system.

As can be seen in the figure the first table TABLE.CO contains three columns the contents of which relate to countries. One column lists countries, a second lists the capitals of the countries, and the third lists the continent to which the countries belong in terms of a continent identity number.

The second table TABLE.EXPORT lists in the first column the name of countries that export various products, and the second column lists which products each country in fact exports.

Finally the third table TABLE.CNT lists relevant continents in one column and a continent identity number in a second column.

A conceptual schema (fig. 2A) contains records of information defining entity types, and records of information identifying relationships between entities. It is created during customization (to be described) and it represents a model which describes the collection of all objects in the information system and all facts about the system, which might be of interest to the users, and the relation between the objects and facts. In other words it is a model of the 'Universe of Discourse' (or object system), which is a selected portion of the real world, or a postulated world dealt with in the system in question.

Thus, a conceptual schema comprises entities (concepts), in the examples denoted as en, where n is an integer, and relationships (links) between these entities (concepts). It has two types of external connections, one to natural language terms (as expressed by natural language vocabulary), and one to data base tables (see EXAMPLES II and IV).

It is very important to recognize that a schema itself is language independent, even though of course the entities may have been assigned "names" expressed in a natural language, e.g. english.

The model as shown in fig. 2, is stored as a set of logical facts:

EXAMPLE I:

```
posesses(e2, e1).
posesses(e2, e5).
posesses(e5, e2).
nom(e6, e2).
acc(e6, e8). // (e6 has-object e8)
subtype(e3, e2).
subtype(e4, e10).
subtype(e5, e7).
identifies(e4, e5). // (e4 identifies e5)
identifies(e10, e7).
name(e11, e7).
lp(e2, e5). //("locative of place"; e2 is-in e5)
```

When customizing the system, the terms likely to be used by the users must be defined. The task of vocabulary definition includes connecting natural language terms to the entities in the schema and providing morphological information on them.

For the data base in our example, the following terms may be defined (the en's are entities in the schema, and the tn's denote the terms; n = integer):

EXAMPLE II:

```
(e1) ---> 'capital' (t1) - noun, plural:
'capitals', pronoun: 'it'
(e2) ---> 'country' (t2) - noun, plural:
'countries', prounoun: 'it'
(e7) ---> 'continent' (t3) - noun, plural:
'continents', pronoun: 'it'
(e8) ---> 'product' (t4) - noun, plural:
'products', pronoun: 'it'
(e6) ---> 'export' (t5) - verb, forms:
'exports', 'exported', 'exported', 'exporting'
(e6) ---> 'produce' (t6) - verb, forms:
'produces', 'produced', 'produced', 'producing'
```

As can be seen the entity e6 has two different natural language terms connected to it, namely 'export' and 'produce'. This signifies that in the object system of the data base, 'export' and 'produce' are synonyms.

The opposite situation could occur as well, e.g. the word 'export' could have the meaning of "the exported products" or it could mean the verb "to sell abroad". In this case clearly the same word relates to two different entities (homonyms).

The customizer can define nouns, verbs and adjectives and connect them to the entities. Note that one entity may be connected to zero, one or several terms in natural language, and that the same term may be connected to more than one entity (concept).

The above definitions are stored as logical facts as a part of the conceptual schema (cf EXAMPLE II):

EXAMPLE III:

```
image(e1, t1).
image(e2, t2).
image(e7, t3).
```

```
image(e8, t4).
image(e6, t5).
image(e6, t6).
category(t1, noun).
category(t2, noun).
category(t3, noun).
category(t4, noun).
category(t5, verb).
category(t6, verb).
term(t1, 'capital')
term(t2, 'country')
term(t3, 'continent')
term(t4, 'product')
term(t5, 'export')
term(t6, 'produce')
syntax(t1, 'capital'.'capitals'.'i'.nil).
syntax(t2, 'country'.'countries'.'i'.nil).
syntax(t4, 'product'.'products'.'i'.nil).
syntax(t3, 'continent'.'continents'.'i'.nil).
syntax(t5, 'export'.exports'.'exported'. 'exported'.'exporting'.nil).
syntax(t6,'produce'.'produces'.'produced'. 'produced'.'producing'.nil).
```

As can be seen, this collection of facts describes the link between the terms and the conceptual schema ("image(...)"), the grammatical identity of terms ("category(...)"), the actual natural language word used for the term ("term(...)"), and the syntax ("syntax(...)") relevant to the term in the language in question (english in this case).

Thus, these expressions define how the terms (tn; n integer) are related to the entities in the schema and what their grammatical identities are.

Dictionary entries are also created during the vocabulary definition. For example, the dictionary entry for the verb 'export' looks like this:

```
verb(verb(18380,feature(typ = na,lg = 1),0),nil,verb__ ('export'))-->'export
```

In order to relate natural language queries to the relational data base, it is necessary to link or connect concepts of the model (i.e. the schema itself) to the data base.

Not all concepts are related to the data base, but there can only be one data base link for a specific concept. Of course several different links may be introduced if necessary, through definition of new concepts.

The links or connections between entities (or concepts) in the schema to the data base are made via SQL expressions:

EXAMPLE IV:

```
(e2) --> SELECT CNTRY FROM TABLE.CO
(e1) --> SELECT CPTL FROM TABLE.CO
(e3) --> SELECT PRDCR FROM TABLE.EXPORT
(e8) --> SELECT PRDCT FROM TABLE.EXPORT
(e11) --> SELECT CNTNNT FROM TABLE.CNT
(e4) --> SELECT CNT ID FROM TABLE.CO
(e10) --> SELECT ID FROM TABLE.CNT
```

The links to the database can be very complicated SQL expressions. The information on such links is stored as the following logical facts and they too constitute a part of the conceptual schema together with the previously mentioned logical facts (see EXAMPLES II and III):

EXAMPLE V:

```
db(e2, set(V1, relation(table.co(V1 = cntry)))).
db(e1, set(V1, relation(table.co(V1 = cptl)))).
db(e3, set(V1, relation(table.export(V1 = prdcr)))).
db(e8, set(V1, relation(table.export(V1 = prdct)))).
db(e11, set(V1, relation(table.cnt(V1 = cntnnt)))).
db(e4, set(V1, relation(table.co(V1 = cnt__id))))
db(e10, set(V1, relation(table.cnt(V1 = id)))).
```

Here "db" indicates the data base link, and "relation" shows the connection between an entity and the corresponding column of a table.

Thus, the conceptual schema consists of a collection of logical facts of the types according to EXAMPLES II, III, and V (Other types could be conceived).

In the following the translation of a natural language query into SQL will be described.

Parsing is the first step in processing a natural language query. The Parser in the Natural Language Analyzer (fig. 1) scans the input string character by character and finds, by using dictionary entries and grammar rules (syntactic rules) in the Analysis Grammar, all possible combinations of patterns which are grammatical. Parsing apparatuses and techniques for syntax analysis are well known in the art and will not be discussed in detail. See for example EP-91317 (Amano, Hirakawa).

The parser produces, as one of its outputs, a parse tree (or syntax tree) or several parse trees (fig. 3A) if the query is ambiguous, describing how dictionary look-ups and application of syntax rules resulted in recognition of an input string as being grammatical.

For example the query 'who exports all products' will generate the parse tree shown in figure 3A (in the APPENDIX some more examples of queries and the intermediate and final structures created in the process are given).

As can be seen in the figure, the top of the tree reads (sent) indicating that the input string was identified as a proper sentence. All joins between branches and the ends of branches are referred to as nodes, having identifiers such as (np), (vc) etc.

The meaning of these identifiers is mostly evident (e.g. (verb), (noun)). However, (np) denotes a 'nominal phrase', (vc) means 'verbal construct' (equivalent to 'verbal phrase'), and (sc) is a 'sentence construct' meaning a grammatically valid clause (not necessarily a complete sentence).

Further, every syntactic rule (grammar rule) is associated with zero, one or more semantic routines (executable programs), and the parser produces as a second output a semantic tree (figure 3B) in association with each syntax tree.

Two examples of grammar rules are given below:

<SENT:1:FPE-COMMAND(1,2)><-<SC:TYP = AZ, + IMP, + CMD,(SYST = 1)__ !(SYST = 2)><NP: + ACC> <SCT:1, + ES, + CN:FPE-NOM(2,1)><-<VC:TYP = NZ, + CNA,COL = COL(2)__ ,-DS,-PPE,-IMP,-PAS,((-SG)&(-SG(2)))!((-PL(2)))><NP: + NOM__ ,-REL,-WPRO>;

These rules are built in one of the many formalisms that exist (in this case ULG), and thus constitute mere examples of how they can be built.

An argument of the syntactic rules may contain a call for or pointer to a semantic routine mentioned above, if appropriate, and for each rule that is activated and contains such "pointer" or "call", a semantic routine is allocated, and a "semantic tree" is built (in the first of the given examples the argument FPE-COMMAND(1,2) is a call for a routine named COMMAND thus building a node named COMMAND; in the second example the argument is a call for NOM).

The semantic trees are nested structures containing the semantic routines, and the trees form executable programs, which produce an intermediate representation form of the query when they are executed.

This intermediate representation form of the original query preserves the meaning of the query, as far as the universe of discourse (or object system) is concerned.

The semantic tree of figure 3B has the following form when expressed as an executable program:

EXAMPLE VI:

```
quest(p01,
  two(p02,
    nom(p03,
      wque(p04, 'who'),
      acc(p05,
        npquan(p06, 'all'
          nomen(p07, 'product')),
        verb(p08, 'export'))))).
```

Here the p's are pointers to the internal structures created during parsing for the input query, and each line begins with the name of the routine called for in the applied syntactic rule.

After completion of the semantic tree the main program enters next loop in which the tree is "decomposed" into its nodes (each individual semantic routine is a node), and the routines are executed from the bottom and up, which will trigger execution of the nested routines in the structure.

The semantic routines "use" the conceptual schema, and the information on the entities in the schema, for checking that the information contents of the generated semantic tree corresponds to a valid relationship structure within the universe of discourse defined by the schema. Thus, the execution of these routines performs a check of a language expression against the conceptual schema to see if the expression is a valid one (within the defined Universe of Discourse or object system).

By using the conceptual schema, the semantic routines generate a representation of the natural language queries in a form called CLF (Conceptual Logical Form). This is a first order predicate logic with set and aggregate functions (such representations can of course be designed in many different ways and still achieve the same object, and the skilled man conceives how this should be done without any inventive work).

The CLF representation of the example query will then be:

EXAMPLE VII:

```
query(
  report,
  set(y1,
    all(y2,
      instance(e8, y2) ->
      exist(y3,
        instance(e6, y3) &
        acc(y3,y2) &
        nom(y3,y1))))).
```

simply meaning that the user wants a report (as opposed to a yes/no answer or a chart) of everything which exports all products and by 'all products' the user can here only mean products appearing as data in the database.

The CLF is then verified, completed, and disambiguated by checking against the conceptual schema. If for example the verb 'export' is defined in the conceptual schema such that it may take subjects from two different entities, then two CLF's must be produced, one for each case. On the other hand if there is no subject for the verb 'export' in the model, the CLF must be aborted.

In the above example, the checking against the model in the conceptual schema results in a more complete CLF as follows:

EXAMPLE VIII:

```
query(
  report,
  set(y1,
    all(y2,
      instance(e8, y2) ->
      exist(y3,
        instance(e6, y3) &
        instance(e3, y1) &
        acc(y3,y2) &
        nom(y3,y1))))).
```

where the added information is that the user wants a list of countries, 'country' (e2) being a supertype of the concept e3, 'producer'.

Contextual references are also resolved at this stage where any reference to previous queries, either in the form of a pronoun or fragment, is replaced by the appropriate CLF statements from those previous queries.

In order to verify the interpretation of the queries with the user and let the user select the correct interpretation among several alternatives generated by the invention, the CLF (Conceptual Logic Form) must be presented in natural language form as paraphrasings of the original query.

To generate natural language from CLF, the CLF first is translated into a set of structures (trees) called Initial Trees. These trees contain such information as what the focus or core of the query is, what concepts are involved in the query, and what are the relationships between them. The following set of Initial Trees will be generated for our example CLF:

noun((id = 3).(group = 1).(scope = nil).var = y1). (entity = e3).(focus = 1).nil).
noun((id = 1).(group = 1).(scope = nil).(var = y2). (entity = e8).(all = 1).nil).
verb((id = 2).(group = 1).(scope = y2.nil).(var = y3). (entity = e6).(acc = y2).(nom = y1).nil).

The paraphrased version of our previous example query will be 'List the countries that export all products'. This paraphrased expression is presented to the user for verification.

When the user has confirmed/selected the interpretation, the corresponding CLF is translated into an SQL-expression. This process involves two steps, namely a translation of the CLF to a further intermediate representation form (Data Base oriented Logical Form; herein referred to as DBLF).

This form is similar to the CLF (or any other equivalent representation that is used), except that the entities are replaced by their data base links from the conceptual schema (see example IV). Thereby the appropriate joins between the SQL tables are established.

In our example, the following DBLF is generated from the corresponding CLF (see example VIII):

EXAMPLE IX:

```
query(
  report,
  set(y1,
    relation(table.co(cntry=y1)) &
    all(y2,
      relation(table.export(prdct=y2)) -->
      relation(table.export(prdcr=y2,cntry=y1)))))
```

The DBLF contains all information necessary to construct the SQL query.

There is also an optimization of the queries by removing redundant join conditions based on the information on the data base elicited during the customization.

If the NL query cannot be translated into one single SQL query, the DBLF will be translated into something beyond pure SQL, and this extension of SQL is called an Answer Set. An Answer Set has the following components:

1) Temporary tables. A query like "How many countries are there in each continent" cannot be represented directly in SQL. To obtain the answer, a temporary table must be created, filled with data and then selected.
The information to do this is part of the Answer Set.

2) Range. There is no range concept in SQL. A query like "List the three highest mountains in the world" cannot be represented. The range specification in the Answer Set takes care of this and it is up to the program displaying the answer to the user to apply it.

3) Report. The third part of the Answer Set is related to how the answer should be presented to the user. There may be three options: Report (default), Chart, or YES/NO.

This makes it possible to handle queries like "Show me, in a bar chart, the sales figures for last month".

For the above example query the following structures will be created:

EXAMPLE X:

```
CREATE TABLE t1 (cntry , card)

INSERT INTO t1 (cntry , card)
  SELECT x1.cntry, COUNT( DISTINCT x1.prdct )
  FROM table.export x1 GROUP BY x1.cntry

SELECT DISTINCT x1.cntry
  FROM table.co x1,t1 x3
  WHERE x1.cntry = x3.cntry
  AND x3.card = (
```

```
SELECT COUNT( DISTINCT x2.prdct )
  FROM table.export x2) .

NIL

REPORT
```

which results in a temporary relation created as the SQL table T1 with the columns CNTRY and CARD. The column CNTRY is copied from the column CNTRY in the table TABLE.EXPORT and the values in the column CARD will be calculated as the number of distinct products (PRDCT column in TABLE.EXPORT) related to each country.

The final query is made against the T1 table and will result in a list of countries which export as many products as the number of distinct products found in the data base - only France in this case.

Each query the user makes is automatically stored in a log. If the query is succesful it is put in a Current Log, and if it fails it is put in an Error Log.

A query in the Current Log may be copied into the input field of the main program. There the user can edit it before it is processed. The Answer Set stored with the query can directly be used to obtain the answer.

The log can be stored and later reused by loading it into a Current Log. It can be viewed in a separate window. Queries appearing in such windows may be copied into the input line and the Answer Set sent to obtain the answer.

There is also provided a facility for creating the conceptual model and the vocabulary definition. This facility is referred to as a Customization Tool.

It is designed to be easy to use by providing a graphic interface (see fig.4), including an editing function, to the person performing the customization (the customizer).

With this interface the following functions are available:

- entities and relationships are presented as symbols (icons)
- the entities and relationships can be manipulated
- the current state of the model under construction is shown by highlighting the objects on the screen in different ways
- sets of objects can be clustered, for hiding complex structures in order to make the model more transparent

The various entity icons 13 used in the graphic interface (see fig. 4) can be e.g. circles, ellipses, hexagons or triangles, whereby the shape is determined by the lexical category of terms referring to the entity in question. Each entity icon is annotated by the entity name.

Relations or sets of relations between entities are represented by line segments (connector icons).

A cluster icon represents a subset of the schema, and has the shape of a rectangle 14.

A small diamond shaped icon (marker icon) is used to represent the current position in the schema.

The Graphic Interface uses the select-then-act protocol to manipulate entities and relationships. Below is given a brief description of the Graphic Interface.

Preferably a mouse is used for ease of use, and a number of options are selectable from various panels and action bars 16. For example 'Create Entity' displays an entity icon in a selected vacant spot on the screen. It also 'opens' the entity for inputting definitions of said entity.

The 'Create Connector' option is operable to create the relationship between two entities. With this option a line segment 15 connecting two previously defined entities is created.

If there are many entities connected to one single 'main' entity, a Cluster can be formed whereby only the selected 'main' entity is displayed, but with a different shape (e.g. a rectangle) to distinguish it from ordinary entity representations.

In a preferred embodiment implemented for a relational data base, the method comprises an initial step of identifying the tables in said data base and defining the relations between the tables. The system then automatically responds by suggesting a conceptual model comprising entities and relationships between these entities. This model is presented to the user (the customizer) for verification.

Thereafter the customizer continues to interactively create entities and relationships in view of his/her

knowledge of the system in question (e.g. a relational data base).

The method also comprises linking the entities to natural language terms, and storing said terms in a dictionary.

The entities are classified as belonging to any of a predfined set of types (person, place, event, process, time, identifier, name etc.), said types being stored.

In addition it comprises creating the links to the data base by identifying which data base representation (e.g. in a subset of SQL; see EXAMPLE IV) the entities shall have.

The whole model including entities, relationships, vocabulary and data base links is stored as (logical) facts.

A still further aspect of the invention is that by keeping knowledge of the system in question and other information used in the natural language analyzing apparatus in data base tables (such as SQL tables), users can use the method and apparatus of the invention to query that knowledge and thus request meta-knowledge.

In this way there is no difference between ordinary queries and meta-knowledge queries, neither from the user's point of view nor from the system's.

The conceptual schema for meta-knowledge is created in advance as a part of a base conceptual schema. Such a schema is application independent, and the tables used for storing said schema are called with unique dummy names when customized. During CLF to DBLF translation (as previously described) when these dummy table names appear in the data base representations, they are replaced with the correct table name corresponding to the current application.

For example, the table where a list of all tables included in the application is kept can be called 'appl tabs' when the schema for meta-knowledge is created. Then, when a specific application 'xyz' is run, the CLF to DBLF translator replaces 'appl tabs' with 'xyz tabs' in the data base representations.

As mentioned previously the conceptual model (schema) is stored as (logical) facts. There are identifiers associated with these facts corresponding to the name of a relational data base table (cf EXAMPLE III where the identifiers are the 'prefixes': 'image', 'category', 'term', etc).

In the process of creating meta-knowledge, when the person doing the customization ends a session, either having completed a model or terminating the modelling temporarily, these facts are automatically read from storage, the identifiers are recognized by the system, and the facts are stored in the empty, predefined tables (linked to the pre-created base conceptual schema). Note that the identifiers are not necessarily identical to the names of the tables; there may be conditions specifying that e.g. the facts belonging to the identifier 'term' be put in a table labled 'words'.

The tables that subsequently are 'filled' with facts are then accessible for querying in the same way as ordinary data base tables, thus providing the desired meta-knowledge.

Claims

1. Natural language analyzing apparatus for use with an information system with a storage comprising a data base containing tables,
an exchangable grammar (8) for a natural language, comprising a set of language dependent rules defining the syntax of said language, whereby certain of the syntax rules have one or more semantic routines associated with them, and
an exchangable vocabulary (6,7) containing definitions of terms of the natural language in question, and morphological information of said terms, the natural language analyzing apparatus comprising,
means (2) for inputting sentences or expressions in natural language,
parsing means (5) which uses the vocabulary (6,7) and the rules in the grammar (8) to check the input sentence or expression for syntactical validity, and which allocates those semantic routines associated with the syntactic rules that were used for parsing, to build one or more executable sets of semantic routines,
generator means for executing the set(s) of semantic routines generated by the parser to create a language independent representation (CLF) of the input,
said storage further containing
i) a set of language independent records of information defining entity types, whereby each entity has a connection to at least one term in the vocabulary, certain entities have a connection to the data base tables, and whereby each term in the vocabulary is an identifier of at least one entity, and
ii) a set of records identifying relationships between different types of entities,
means for producing from verified ones of said langauge independent representations, a natural language output indicative of the systems interpretation of the input, and for requesting confirmation of correctness of

the interpretation, and
means responsive to confirmation for producing a database query from the verified language independent representation.

2. Apparatus as claimed in claim 1, wherein the data base is a relational data base (DB).

3. Apparatus as claimed in any preceeding claim, comprising means for storing previous queries and corresponding answer sets, said answer sets comprising guery statements, a specification of how much of the data in the data base tables is to be presented to the user, and information on the mode of presentation of the data.

4. Method of analyzing natural language in a computer information system with a storage comprising a data base containing tables, an exchangable grammar (8) for a natural language, the grammar comprising a set of language dependent rules defining the syntax of said language, whereby certain of the syntactic rules have one or more semantic routines associated with them, and an exchangable vocabulary (6,7) containing definitions of terms and morphological information about said terms, of the natural language in question, a first set of language independent records of information defining entity types, whereby each entity has a connection to at least one term in the vocabulary, certain entities have a connection to the data base tables, and whereby each term in the vocabulary is an identifier of at least one entity, and a second set of records identifying relationships between different types of entities, the method comprising the steps of
parsing an input expression for generating one or more syntactically valid interpretations of said input, by checking the input against terms in the vocabulary, and against the syntax rules in the grammar, some of which rules contain pointers to or a call for a semantic routine,
as a response to using a rule containing a said pointer, allocating a said semantic routine, and building one executable set of such semantic routines for each syntactically valid interpretation of the input, the sets being nested structures,
executing each of said nested structures for generating a language independent intermediate representation form of the natural language input, and during said execution,
checking said records of information relating to entities occuring in the input to determine the validity of the information contents of the nested structure, corresponding to said syntactically valid interpretation of the input,
aborting invalid interpretations and producing from valid interpretations a natural language output indicative of the systems interpretation of the input and requesting confirmation of the validity of the interpretation, and
as a response to confirmation, generating from said intermediate representation, a query to the database.

5. Method as claimed in claim 4, wherein if an answer cannot directly be retreived from the data base tables in one single query statement, temporary tables are created and filled with data, said temporary tables being queried for the final answer.

6. Method as claimed in claim 5, wherein the data is orderd in an ascending or descending order.

7. Method as claimed in claim 6, wherein only a selected portion of the data is presented to the user.

8. Method as claimed in claim 7, wherein an answer set comprising an instruction to create and fill the temporary tables, together with the query and the range of data to be selected from the temporary tables, is stored in a log for later use.

9. Method as claimed in claim 8, wherein a stored answer set is used by copying a stored query into an input field of a query panel of a query program.

10. Method of creating a conceptual model of a data base by means of a graphic interface to said data base, said conceptual model comprising entities and relationships between entities, corresponding to the data in said data base, said graphic interface providing display means having an action bar listing a number of action options, and a graphic area, the method comprising the steps of

a) creating an entity representation on the screen by selecting from the action bar an option 'create entity', whereby an entity icon, and a dialog box for entering the data of said entity are displayed,

b) creating an identifier indicative to which natural language term the entity relates by entering said term and synonyms thereof in an selected input field of the box, entering the category of the term, and entering syntax information about the term, whereby the entered information is stored as logical facts,

c) classifying the entity as belonging to any of predefined set of types, by selecting a type name from a panel, whereby said type name is stored,

d) creating a relationship between two entities by selecting an option 'create connector' from the action bar, identifying which two entities are to be connected, and entering the appropriate relationship as a selection from a number of displayed predetermined possibilities, whereby the relationship is stored as a fact,

e) creating a connection between entities and tables in the data base by entering in an appropriate

data field in the dialog box, which data base representation the entity shall have, and

f) repeating steps a) to e) until a satisfactory model has been created.

11. Method as claimed in claim 10, further comprising an initial step of identifying the tables in the data base, and defining the relations between said tables, whereby entities and relationships between entities of an initial conceptual model automatically is suggested and presented to a user for verification or editing.

12. Method as claimed in any of claims 10 or 11, comprising creating dictionary entries simultaneously with the term definition in step b).

13. Method as claimed in any of claims 10 to 12, wherein optionally entities are clustered under a common name by selecting a 'cluster' option from the action bar.

14. Method of creating meta-knowledge of a computer resident data base during conceptual modelling thereof, wherein the model that is created contains entities and relationships between entities, corresponding to the data in the data base, said model being stored as facts containing identifiers corresponding to names of empty and predefined meta-knowledge tables in a relational data base, the method comprising automatically reading said facts from storage and storing the facts belonging to a certain identifier in a corresponding empty and predefined meta-knowledge table in said relational data base.

15. Method as claimed in claim 14, wherein the predefined tables have application independent names which are changed when a specific application is run.

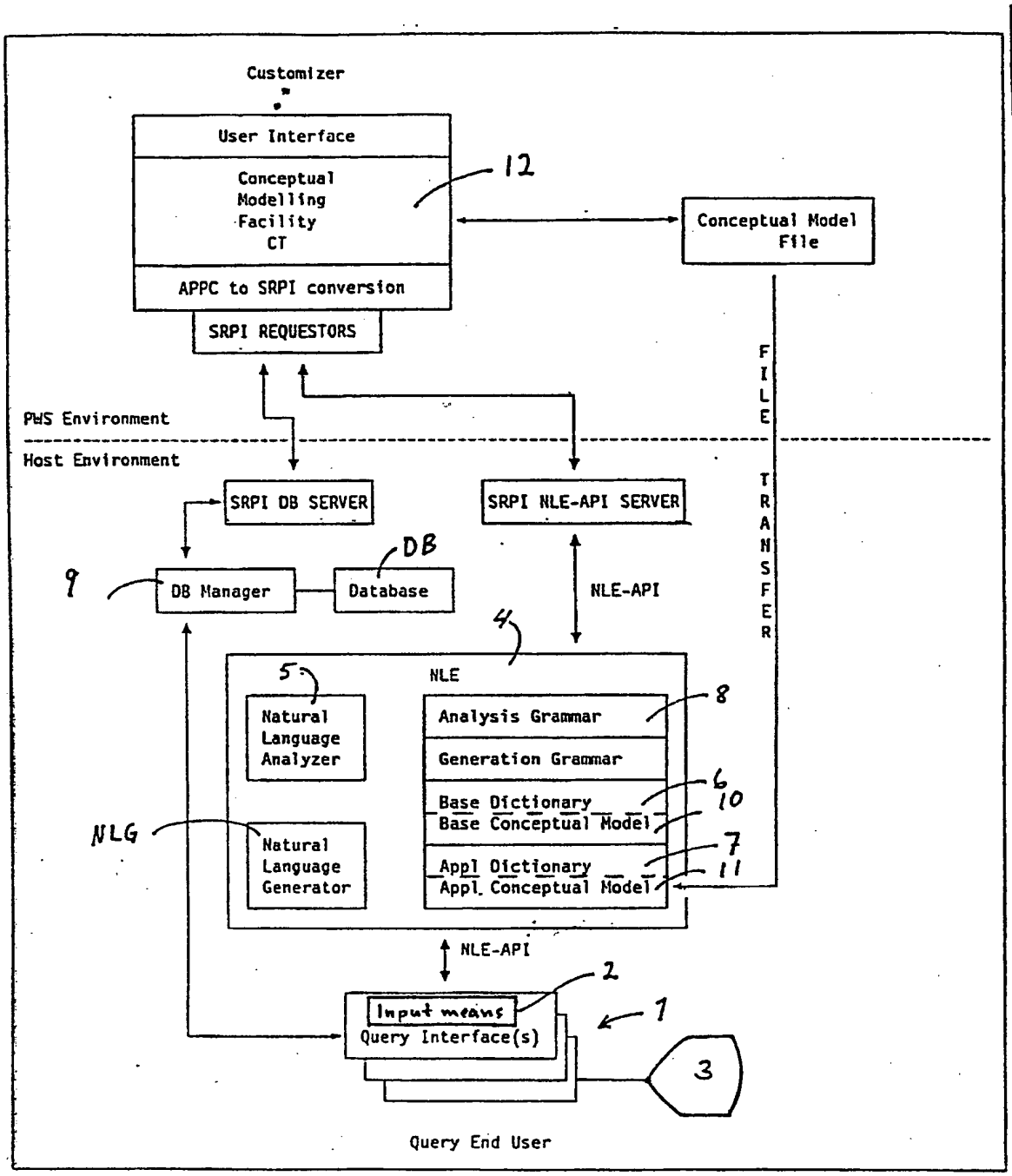
Customizer
User Interface
Conceptual Modelling Facility CT
12
APPC to SRPI conversion
SRPI REQUESTORS
Conceptual Model File
FILE TRANSFER
PWS Environment
Host Environment
SRPI DB SERVER
SRPI NLE-API SERVER
DB
9
DB Manager
Database
NLE-API
4
5
NLE
Natural Language Analyzer
Analysis Grammar
8
Generation Grammar
6
Base Dictionary
10
Base Conceptual Model
NLG
Natural Language Generator
7
Appl Dictionary
11
Appl. Conceptual Model
NLE-API
2
Input means
Query Interface(s)
1
3
Query End User

Fig. 1

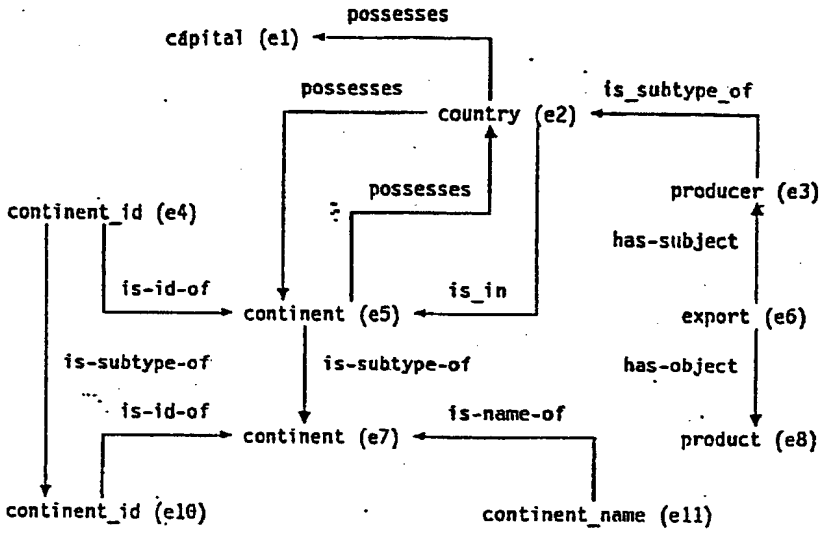
possesses
capital (e1)
possesses
country (e2)
is_subtype_of
possesses
producer (e3)
continent_id (e4)
has-subject
is-id-of
is_in
continent (e5)
export (e6)
is-subtype-of
is-subtype-of
has-object
is-id-of
is-name-of
continent (e7)
product (e8)
continent_id (e10)
continent_name (e11)

Fig 2A

QUESTION: 'WHO EXPORTS ALL PRODUCTS?'

TERMS NEEDED IN MODEL

ENTITIES AND RELATIONSHIPS

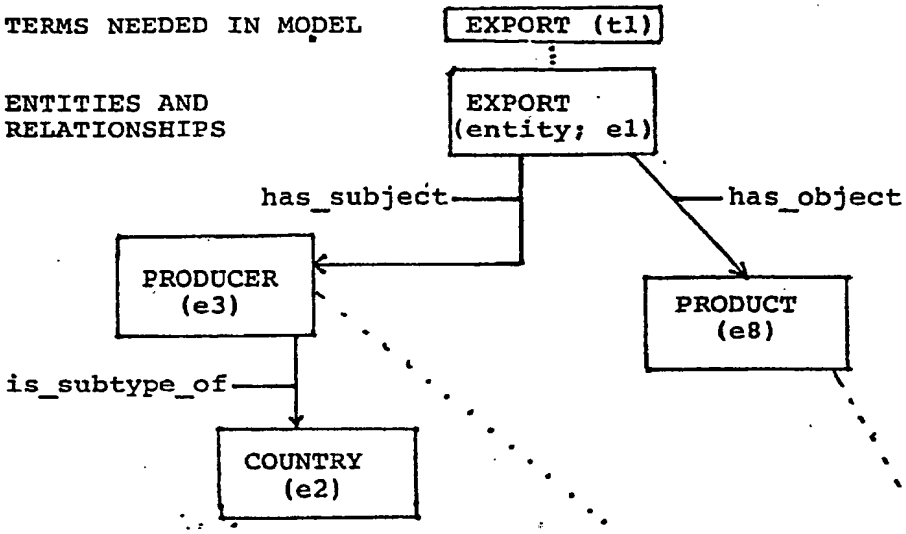

TABLE.CO

CNTRY	CPTL	CNT_ID
FRANCE	PARIS	5
JAPAN	TOKYO	3
BRAZIL	BRASILIA	2
SWEDEN	STOCKHOLM	5

TABLE.EXPORT

PRDCR	PRDCT
JAPAN	CAR
JAPAN	TEXTILE
SWEDEN	STEEL
SWEDEN	TEXTILE
FRANCE	WINE
FRANCE	CAR
FRANCE	STEEL
FRANCE	TEXTILE

TABLE.CNT

CNTNNT	ID
AFRICA	1
AMERICA	2
ASIA	3
AUSTRALIA	4
EUROPE	5

Fig 2B

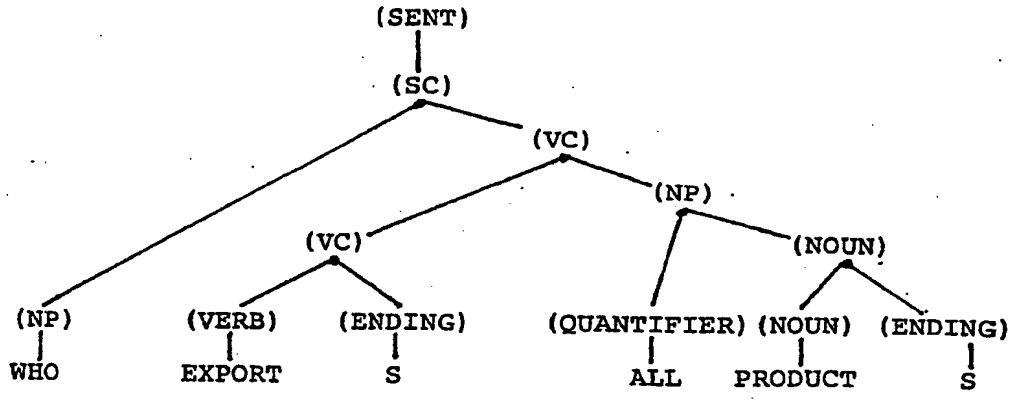
(SENT)
(SC)
(VC)
(NP)
(VC)
(NOUN)
(NP)
(VERB)
(ENDING)
(QUANTIFIER)
(NOUN)
(ENDING)
WHO
EXPORT
S
ALL
PRODUCT
S

Fig 3A

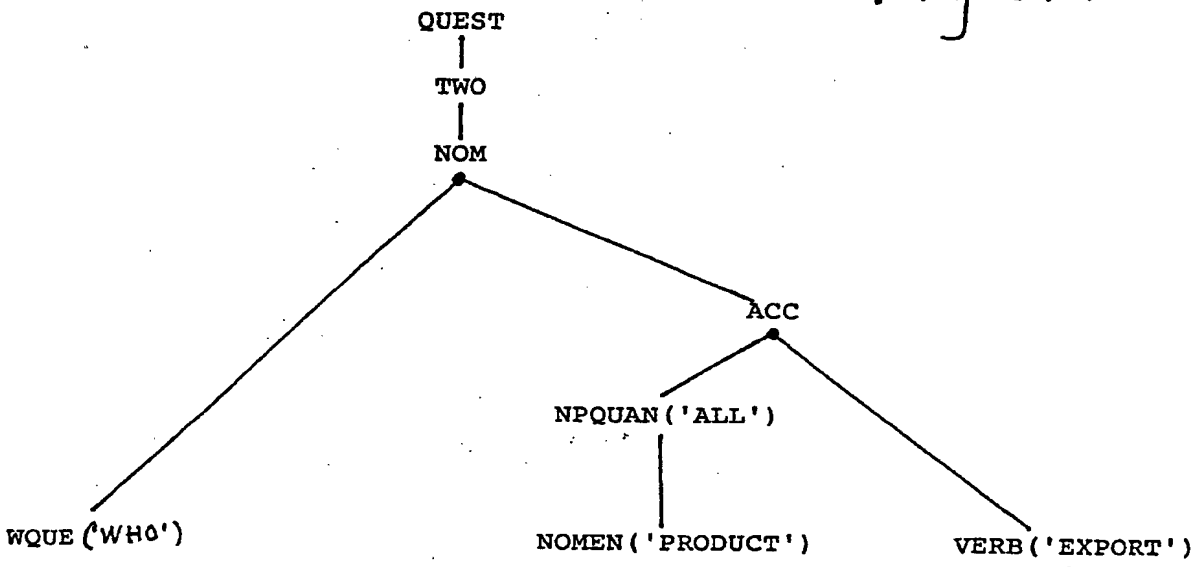
QUEST
TWO
NOM
ACC
NPQUAN('ALL')
WQUE('WHO')
NOMEN('PRODUCT')
VERB('EXPORT')

Fig 3B

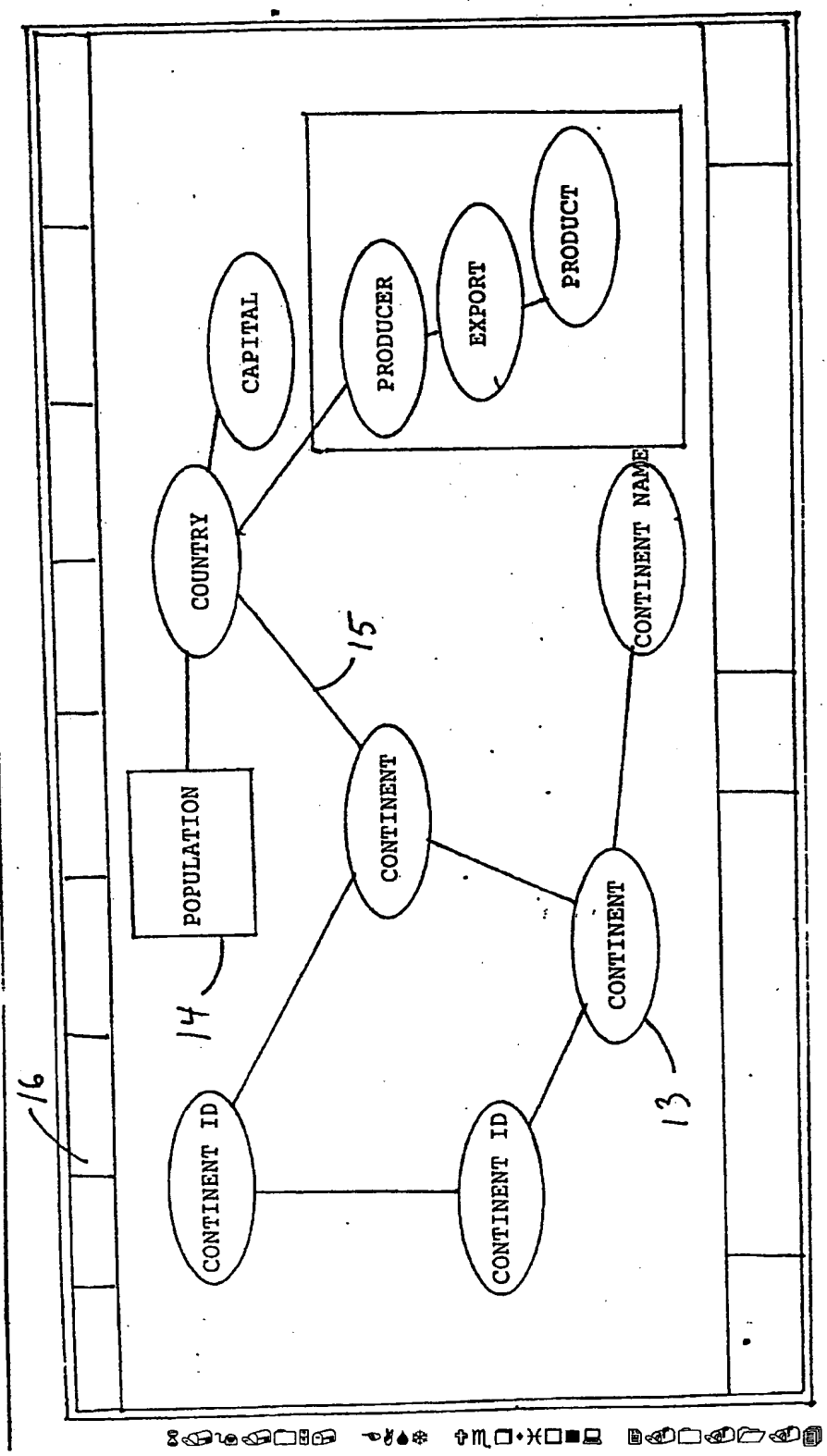
CAPITAL
PRODUCER
EXPORT
PRODUCT
COUNTRY
CONTINENT NAME
15
POPULATION
CONTINENT
CONTINENT
14
16
CONTINENT ID
CONTINENT ID
13

Fig. 4

European Patent Office

EUROPEAN SEARCH REPORT

Application number
90850095.2

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.4)
A	US-A- 4 736 296 (Y. KATAYAMA ET AL)	1-15	G 06 F 15/38 15/40
A,D	GB-A- 2 096 374 (MARCONI COMPANY)	1-15	
A,D	EP-A2- 168 814 (NEC CORPORATION)	1-15	
A,D	US-A- 4 688 195 (C.W. THOMPSON ET AL)	1-15	

TECHNICAL FIELDS SEARCHED (Int. Cl.4)

G 06 F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
STOCKHOLM	21-05-1990	SILFVERLING J.

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO Form 1503 03.82